


Fully connected Autoencoder

Deadline: 30 Oct 23:59 

Bonus

The goal of this problem is to understand the internal bottleneck representation that a simple autoencoder uses to encode information. The MNIST digits dataset is used in this problem set.

To obtain credits for this exercise, upload a PDF-document presenting your results (details below). Use the upload button at the top of this page. Merge the PDF-files you have uploaded for Homework 3 into one document. Make a front page, and attach the computer code you have used in appendices. Submit your merged PDF-document to URKUND, before the deadline.

Description and Data Preprocessing

An autoencoder is a neural network that is trained to reproduce its inputs and in the process learns an internal representation of the dataset. In this exercise we analyse the internal representation of an autoencoder by implementing simple fully connected autoencoders with a bottleneck layer. The autoencoder learns a compressed representation of the dataset due to the presence of the bottleneck layer, which has a much smaller dimensionality than the input layer.

Load the MNIST dataset (you can use the provided helper file, LoadMNIST.m, with argument "3"). Transform the 28×28 image matrices into vectors of length 784. Normalize the elements of the vectors so that they are restricted between 0 and 1.

Autoencoder 1: Construct a fully connected neural network consisting of three fully connected layers, each followed by a ReLU layer. The precise layout is as follows:

- Sequence input layer.
- Fully connected layer with 50 units (set the "WeightsInitializer" argument to "glorot"), followed by a ReLU layer.
- Fully connected layer with 2 units (set the "WeightsInitializer" argument to "glorot"), followed by a ReLU layer. *This is the bottleneck layer.*
- Fully connected layer with 784 units (set the "WeightsInitializer" argument to "glorot"), followed by a ReLU layer.
- Regression layer.

Autoencoder 2: This is the same as the previous autoencoder, except use 4 units in the bottleneck layer.

Training: Use the normalised 784 length vectors both as inputs and targets for training. Train both networks using the adam optimiser with MiniBatchSize = 8192 and InitialLearningRate = 0.001. Shuffle the training set before each epoch. Train the network for at least 800 epochs. *Training tips:* Use the GPU execution environment if possible (MATLAB: 'ExecutionEnvironment', 'gpu') for shorter training time. It is possible to stop training and resume at a later time by saving the network, and reloading before the next training (MATLAB: use the "save" command to save the network in a .mat file, in order to resume training for a network, "net", use "net.Layers" as an argument for the function "trainNetwork", [net, tr] = trainNetwork(xTr, xTr, net.Layers, options)).

Analysis

- In order to understand whether the networks have trained sufficiently, compare the input and output images as a montage (original on the left, network output on the right) for different digits. *Include in your solution such a montage, one for each autoencoder, that shows one of each digit. State how many digits each autoencoder convincingly reproduces.*
- Divide each trained network into an encoder and decoder. For this you will make two new sets of layers, layers_encode, layers_decode, which contain the appropriate layers of net.Layers. For example, for the encoder, set layers_encode(1)= net.Layers(1), and so on until you reach the bottleneck layer. Set the final layer as a regression layer. Use the "assembleNetwork" command to make a new network, net_encode = assembleNetwork(layers_encode).
- For autoencoder 1, make a scatter plot of the output of the bottleneck neurons using the training dataset. Use one colour for each digit that is well reproduced (as visible in the montage), and ignore the digits that are not well reproduced. Do you observe a pattern? Which rule does the encoder use to represent the digits? Test the rule you found by feeding it to the decoder. Does it reproduce the required digits? Next add also digits that are not well reproduced by the autoencoder to this scatter plot. Restrict yourselves to the first 1000 samples of the test dataset, in order to avoid overcrowding the plot. *Include this final scatter plot in your solution. The well-reproduced digits must be clearly visible.*
- For autoencoder 2, find the coding rule by inspection using the well-classified digits. Feed the code for a digit into the decoder. Does it reproduce the expected digit? *Include a two-line description of the rule used by the autoencoder in your solution. Also include one table showing the input code to the decoder on the left, and the output image on the right for autoencoder 2. Please make one row for each digit that is well reproduced by the autoencoder. Discuss in one sentence what happens for the other digits, which are not well reproduced.*

Submit at most one A4 page with 12pt single-spaced text, and with 2cm margins. Each page may contain at most one Figure and/or one Table with the corresponding Figure and/or Table caption, in addition to the text discussing the results shown in the Figure/Table. It is not necessary to write a full page for each of the four problems in homework 3, but you must explain/describe what you have done and clearly state your answers/results to the questions, as well as your conclusions. When necessary you must discuss possible errors and inaccuracies in your results. Plots/graphs must have legible axis labels and tic labels. All

symbols and lines must be explained in the Figure or in a caption. The Figure may consist of separate panels, label them 'a', 'b', and so forth.